OCCUPANCY GRID MAP MERGING FOR MULTIPLE ROBOT SIMULTANEOUS LOCALIZATION AND MAPPING

S. Saeedi*, L. Paull*, M. Trentini†, and H. Li*

Abstract

In robotics, the key requirement for achieving autonomy is to provide robots with the ability to accurately map an environment and simultaneously localize themselves within that environment. This problem is referred to as Simultaneous Localization and Mapping (SLAM). In this research, a decentralized platform for SLAM with multiple robots has been developed. Single-robot SLAM is achieved through Extended Kalman Filter (EKF) data fusion. This approach is then extended to multiple-robot SLAM with a novel occupancy grid map fusion algorithm.

Map fusion is achieved through a multi-step process that includes image pre-processing, segmentation, cross correlation, approximating the relative transformation matrix, tuning of the transformation matrix, and finally verification of the result. Results are shown from tests performed in real-world environments with multiple homogeneous robotic platforms. [1]

Keywords: Simultaneous Localization and Mapping (SLAM), Multiple Robots, Map Merging, Segmentation, Radon Transform, Image Entropy.

*COBRA Group at the University of New Brunswick, Fredericton, Canada, http://www.ece.unb.ca/COBRA/, {sajad.saeedi.g, liam.paull, howard}@unb.ca

†Defence Research and Development Canada-Suffield, Alberta, Canada, Mike.Trentini@drdc-rddc.gc.ca

[1]Preliminary results of this work have been presented in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2011 [1].

1 Introduction

The first step required to provide any kind of robotic autonomy is for a robot agent to be able to build a map of its environment and localize itself within it. It cannot be assumed that the agent will have a map of the environment, so the mapping and localization tasks must occur simultaneously. This process is referred to as Simultaneous Localization and Mapping (SLAM). The central goal of this research is to develop a unified structure for solving the SLAM map merging problem for multiple robots.

SLAM is achieved through the process of mapping out an unknown environment and generating a consistent map by fusing sensory information. Typical sensors used include encoders for wheeled robots, Inertial Measurement Units (IMU), cameras [29], and laser rangefinders [31] among others. Using SLAM, the position of the robot can be updated and the environmental map can be built and maintained [37].

Numerous methods have been proposed for SLAM but generally they can be divided into two groups:

- **Feature-based SLAM**: Uses extracted features or landmarks in the environment and keeps a list of them as part of the map [30], [14]. As a result this method requires feature extraction and is therefore limited to environments with features. Most feature-based solutions use vision sensors [8], underwater sonar [19], [24], or occasionally LiDAR [3].

- **View-based and appearance-based SLAM**: Are based on raw sensor data processing techniques. View-based SLAM usually requires a laser rangefinder (LiDAR) [13], [15] while appearance-based SLAM is achieved with a camera [35]. In view-based SLAM, scan matching [23], [26], [6] is used for the data association of the laser measurements and the map is represented in occupancy grid format [32], [9]. In appearance-based SLAM the appearance of the current view is compared with the available views (reference images) to localize the robot [35], [20].

The majority of past literature has focused on SLAM with one sensing platform. Using multiple robots for SLAM has the advantage that exploration and mapping tasks can be done faster and more accurately. In addition, a distributed system is more robust since the failure of any one of the robots does not halt the entire mission [4]. Collaboration based operations such as fire fighting in forest and urban areas, rescue operations in natural disasters, cleaning marine oil spills, underwater and space exploration, security, surveillance and maintenance investigations need to be completed quickly and autonomously and require localization and mapping.

Having multiple robots adds a new layer of challenges to the SLAM problem. In a multi-robot framework, each agent should achieve its own goals, which are to provide an accurate estimation of its surroundings and current position, while contributing to the global coordinated tasks. This requires that each robot react to information obtained from other robots, and also incorporate new information into its decision structure to react to environmental changes. An increase in the number of robots will increase the time and space complexity as well as the reliance on reliable communications. It should be avoided that one robot performing poor mapping adversely affects the maps of others.

Sharing data among robots is a fundamental issue in multiple-robot SLAM. Past approaches to collaborative SLAM can generally be categorized based on whether they share raw sensor data [18] or processed data [4]. Raw sensor data means that sensed information such as laser ranger measurements and wheel odometry readings are not processed. In processed data, such as a map or poses of robots, sensor readings are processed through filtering or smoothing or other methods. Sharing raw sensor data results in more flexibility but requires high bandwidth and reliable communication between robots as well as more processing power. In contrast, sharing maps uses less bandwidth and there is no need to process raw data redundantly; however, the performance is dependent on the quality of maps. The latter method is usually referred to as map merging or map fusion. The works by Thrun [39] and Howard [18] are examples of sharing raw laser and odometry data among robots. The works by Carpin et al. [5] and Birk et al. [4] are examples of sharing processed maps between robots. The choice of the method depends on several factors such as the application and the available resources.

It should be emphasized that the accuracy of any multiple robot SLAM solution which is based on the map merging, depends on the accuracy of the individual local maps of the robots. Therefore to have a better and consistent global map, it is important to use a robust SLAM solution to generate local maps. The proposed method is a map merging solution, and unlike filtering methods, it does not cause accumulated errors. In the proposed method, similar map segments from both maps are processed to calculate the relative transformation of the maps. Then using this transformation, a global map is generated.

In [34], a feature-based multiple robot SLAM algorithm is proposed which uses an information filter. A feature-based multiple robot localization is proposed in [22] where fuzzy sets are used to represent uncertain position information and fuzzy intersection [28] is used to fuse data. In [38] an Extended Kalman Filter (EKF)-based solution is proposed where features and poses are filtered by Kalman filtering. In [12] feature-based visual SLAM using particle filtering is proposed. In multi-agent systems, communication limitation is another important issue. In [2] an algorithm for merging feature-based maps with limited communication is introduced. In [21] feature-based map merging for multiple robots under limited communication is proposed where determining and extracting good features remain as future work.

While the feature-based paradigm has been shown to be efficient and effective in certain environments, in general this is not the case mainly due to the loss of the data in the representation of the map as a set of features and its dependency on the quality of the feature extraction algorithm[33]. With the available powerful onboard processing capabilities, it is possible to use view-based SLAM for multiple robot SLAM and avoid the loss of information.

In the method proposed in [18], raw sensor measurements are shared among robots. Particle filtering is used for data fusion and it is assumed that the robots will meet each other at a point. At the meeting point, the robots determine their relative positions and can work backwards to fuse all the previous raw data. The assumption that the robots can meet at a point is the main drawback of the solution. In [4] a solution is presented based on occupancy map merging. This method uses map-distance as a similarity index and tries to find similar patterns in two maps based on a random walk algorithm. The main drawback of this method is that it usually fails especially when there are fewer overlaps between the maps. The work in [11] is based on [4], but at the single robot level, Distributed Particle SLAM (DP-SLAM) version 2.0 [10] is used. The method proposed in [4] and used in [11] does not scale well with the size of the environment. A similar method is proposed in [5] with simulated annealing and hill climbing used to merge maps. This method also becomes ineffective in maps with less overlaps.

Map merging requires finding the overlap between maps without any *a priori* knowledge about the robots' relative positions. Given two or more occupancy grid maps, overlaps can be identified by extracting certain features from maps and comparing them. These features are grid map features (such as geometric shapes or combinations of lines) and should not be confused with features used in the feature-based SLAM. For example, two parallel straight walls, representing a corridor, can be considered an occupancy grid feature. Once these common features are identified, it is possible to use them for finding the relative transformation between maps. Different approaches are available to extract features like line segments from maps and then compare them [16], [25]. The Hough transform [7] and the Radon transform [27] are two classical methods for line extraction which have different applications. In [25] different solutions of line extraction such as Split-Merge and the Hough transform are compared.

In this research, the proposed solution to merge occupancy grid maps is based on matching segments in the maps that are not straight lines. Featured segments carry more information than straight lines making the matching process more robust since unique features are less likely to be incorrectly matched. This novel map feature extraction is based on processing histograms of map segments. In a typical grid map which is full of similar right angle corners and straight walls, it can robustly identify unique map features and use them for map fusion.

In the mapping process, uncertainty in system modeling, measurement and processing causes noise accumulation in the grids of the map. For example, a single occupied point (cell) in the real world can cause many surrounding cells to appear occupied because of the sensor and process uncertainty. As a result, more processing power will be required to handle the falsely detected occupied cells. As a contribution of this paper, to solve this problem, edges of obstacles are extracted by the Canny edge detector algorithm, then, since the artificial thickness of the obstacles caused by sensor noise will result in multiple edges being produced, a novel edge smoothing method is applied to remove redundant edges.

To find overlaps between the smoothed edge maps, usually an exhaustive search method is required over the whole domain of the maps. As another contribution of this paper, map features are extracted and processed to reduce the domain of the search. Most map feature extraction methods like the Split-Merge, the Radon and the Hough transform algorithms rely only on straight lines and ignore featured segments. In this paper a novel map feature extraction is introduced that analyzes the histogram of occupied map segments. If the histogram of a segment satisfies specific criteria, it is accepted as a featured segment of the map.

The next step is to determine the transformation using the extracted features. First an approximate transformation is determined by matching features, then the results of the transformation are tuned to provide a better transformation.

The result is a multi-step robust and accurate map merging algorithm that is capable of merging maps with minimal overlap from multiple robots without knowing their relative poses.

The rest of the paper is organized as follows: The next section will present a more detailed problem statement. Section 3 introduces the proposed method for multiple robot SLAM, Section 4 presents some experimental results, and Section 5 makes some general conclusions and discusses future work.

2 Problem Statement

In this section, formal definitions and requirements for the multiple robot SLAM are presented. First single robot SLAM is introduced mathematically, then the occupancy grid map is defined and finally the map merging problem is discussed.

2.1 Single Robot SLAM

The mathematical model of a mobile robot is presented as

$$x_t = f(x_{t-1}, u_t) + r_t \tag{1}$$

$$z_t = h(x_t) + v_t, \tag{2}$$

where $f(\cdot)$ and $h(\cdot)$ are the nonlinear process and observation models. x_t is the 2D pose of the robot at time t which includes 2D Cartesian coordinates and the orientation of the robot. u_t is the control signal which for ground robots could be determined from wheel encoders or an IMU. r_t and v_t are known Gaussian process and observation noises. The state of the system is filtered by an Extended Kalman Filter (EKF) [33]. The data association of the EKF-based SLAM which provides observation signals from sensors and relates different observation at different times is called scan matching [23], [6].

Given two scans at times t and $t-1$, scan matching provides the relative transformation between the two scans. The filtered state together with laser measurements of range from obstacles are used to generate an occupancy grid map representation of the world.

2.2 Occupancy Grid Mapping

Occupancy grid mapping is a standard method for mapping using range measurements. A map m can be represented as a set of N grid cells, each with an associated binary random variable, representing its occupancy:

$$m = \{m_i\}, i = 1, ..., N. \tag{3}$$

The binary random variable specifies whether a cell is occupied ('1') or free ('0'), then $p(m_i = 1)$ or $p(m_i)$ represents the probability that the cell is occupied.

In an occupancy grid map, it is desired to calculate the posterior over the map given the trajectory of the robot and available measurements as follows [33]

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \tag{4}$$

The factorization defined in (4) makes updating the map tractable.

Assuming initially a cell i has *unknown* occupancy, $p(m_i) = 0.5$, at time t, if the cell is in the perception field of the range sensor, its value is calculated using the following recursive relation:

$$l_{t,i} = l_{t-1,i} + \log \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)}. \tag{5}$$

where $l_{t,i}$ is referred to as the log odds and is defined as

$$l_{t,i} = \log \frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})}, \tag{6}$$

The *log odds* representation avoids numerical instabilities for probabilities near zero or one. The probability of $p(m_i|z_t, x_t)$ is sensor specific and for a laser rangefinder can be modeled using the approach presented in [33].

2.3 Map Merging

Let a transformation be composed of a 2×2 rotation matrix R_θ and a 2×1 translation vector T as follows

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, T = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}, \tag{7}$$

Let m_1 and m_2 be two occupancy grid maps. Assuming that m_2 is merged into m_1, the map merging problem is defined as: find a rotation matrix, R_θ and a translation vector, T which transforms m_2 such that the overlaps of m_1 and the transformed m_2, m_2', fall squarely on top of each other. To do this it is required to maximize a verification index defined on the merged map:

$$(R_\theta, T) = \underset{\theta,\delta_x,\delta_y}{\operatorname{argmax}} V(m_1, m_2'), \tag{8}$$

where m_2' means that the m_2 is transformed according to R_θ and T. $V(\cdot)$ is a criterion that evaluates the merging process and will be explained in subsequent sections.

This optimization is not easy to solve analytically and most methods [5], [4] use an exhaustive search to find a transformation matrix and then check (8) to see if it is satisfied. This paper proposes a novel approach that uses image processing techniques to find a good estimate of the optimized R_θ and T very efficiently, therefore greatly reducing the domain of the search and the computational effort required.

Fig. 1 depicts an example of map merging with three maps. The overlap between map_1 and map_2 is shown with a dashed ellipse and the overlap between map_1 and

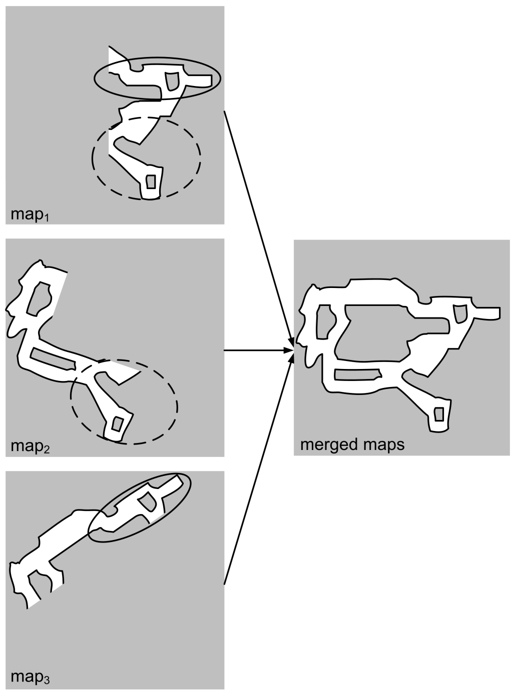

Figure 1: An example of map merging. Three maps are merged to form a complete map of the environment. The overlap between map_1 and map_2 is shown with a dashed ellipse. The overlap between map_1 and map_3 is shown by a solid line ellipse.

map_3 is shown by a solid line ellipse. To find the required transformation, the first step is to identify overlaps, then use the overlaps to calculate the alignment and finally verify the results using (8).

The map merging problem can be interpreted as a special case of image registration in computer vision where images are maps with special geometry. However, the problem is that in multiple robot map merging, the percentage of overlap between maps is usually low.

After finding the relative transformation between two maps, m_1 and m_2, the occupancy grid map probabilities are combined to produce the final map. The data that is received from the transformed map, m_2' is akin to a batch of sensor data and should be incorporated by using the additive property of the log odds representation of occupancy originally defined in (6):

$$l_{t,i}^{fused} = l_{t,i}^{1} + l_{t,i}^{\prime 2} \tag{9}$$

for all $i = 1..N$. Where $l_{t,i}^{1}$ and $l_{t,i}^{\prime 2}$ are defined in (6). The superscript indices are used to identify the maps, l^1 is for m_1 and l'^2 is for m'_2. l^{fused} represents the fused map, m_{fused}.

3 Proposed Method

As mentioned, multiple robot SLAM has the advantage that it is more efficient and robust to robot failure at the cost of increased development and computational complexity. In multiple robot SLAM, the map provided by each robot in its own reference coordinates is called the local map. Each local map is generated from coordinated laser scans. Each robot tries to integrate all of the local maps provided by the other robots to generate a global map of the environment. However, this is a difficult task because the required alignments or transformation matrices which relate these maps to one another are unknown. The misalignment of the maps is caused by the fact that the relative poses between the robots are unknown. As a result, once the map fusion process has successfully found a transformation between the robots, it can be used to merge the local maps from that point onwards.

3.1 Overview of Proposed Method

The proposed map fusion algorithm in this paper is based on a search and verification algorithm. Fig. 2 shows the algorithm for two robots, $robot_1$ and $robot_2$ with unknown relative positions and each with its own local map, map_1 and map_2 respectively.

This algorithm is scalable and can be used for more than two robots. Maps in this algorithm are assumed to be in the form of the occupancy grid maps [33].

According to Fig. 2, the diagram of the system, the inputs are map_1 and map_2.

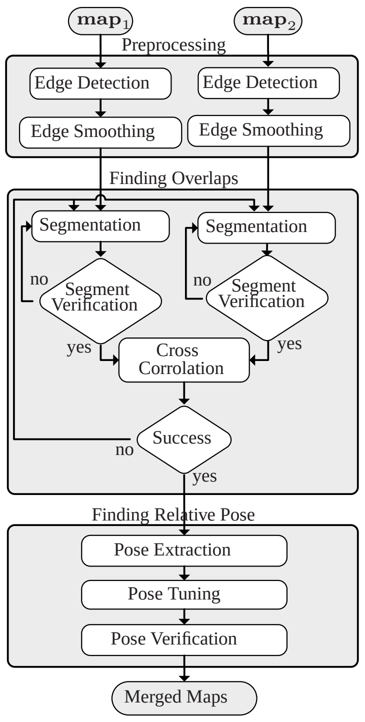

Figure 2: The proposed map fusion algorithm. Two input maps, $\mathbf{map}_1$ and $\mathbf{map}_2$, are fused by finding their relative transformation matrix. No prior information is available regarding the relative position of two respective robots.

The algorithm is composed of three main steps:

- **Preprocessing**: In this step maps are processed to reduce the computational demand in the next steps and obstacles are represented in an abstract form.
- **Finding overlaps**: Common parts between maps are extracted in this step. These overlaps are used in the next step.
- **Finding relative pose**: In this step, overlaps are used to calculate the transformation matrix. The results are tuned and verified in this step.

First, in the **Edge Detection** block, the Canny edges of the maps are extracted. Then, in the **Edge Smoothing** block, the extracted edges are smoothed to facilitate processing in the subsequent blocks. In the **Segmentation** block, a segment of obstacles in the smoothed map is selected. The segment is passed through the **Segment Verification** block to ensure that it contains enough unique geometric information to be used for comparison. The segmentation and verification process continues until an acceptable segment is found or a maximum number of search iterations is reached. The same process is done on map_2. If acceptable segments from both maps are found then the selected segments are tested for similarity in the **Cross Correlation** block. If the selected segments are deemed to be congruent, then they are passed to the **Pose Extraction** block where a rough estimate of the relative pose of the robots is determined. In the **Pose Tuning** block, the orientation of the approximate relative pose is further tuned using the Radon transform [27] and then the translation elements of the relative pose are tuned by a similarity index. In the **Pose Verification** block, the final results are verified. If pose verification rejects the calculated transformation, then the process can start over with other segments generated in the **Segmentation** block or at a later time when maps have more overlaps. This is not shown in the block diagram to keep it simple. In the following sections, each block is explained in detail.

3.2 Edge Detection

The first processing block is **Edge Detection**. Here, noisy measurements are removed from both maps using a smoothing filter and then edges are detected using the Canny edge detection method [36]. Canny edge detection involves the following steps: 1) The input image is filtered by convolving it with a Gaussian filter, 2) The gradient of the image, J is calculated and has components along the horizontal and vertical axes as given by:

$$\boldsymbol{J} = J_x\hat{i} + J_y\hat{j}. \tag{10}$$

3) From the derivatives, the edge strength and direction can be determined by:

$$\boldsymbol{E}_s = \sqrt{J_x^2 + J_y^2}, \tag{11}$$

$$\boldsymbol{E}_o = \arctan(\frac{J_y}{J_x}). \tag{12}$$

4) Non-maximum suppression is performed on the map. In non-maximum suppression, cells whose values of $\boldsymbol{E}_s$ are not larger than their neighbors along the direction perpendicular to the edge orientation in $\boldsymbol{E}_o$ are set to zero. 5) A hysteresis thresholding is performed: given a high threshold t_h and a low threshold $t_l(t_h \geq t_l)$ cells are marked as edges if either: 1) $E_s \geq t_h$ or 2) $E_s \geq t_l$ and connected to an established edge point $\hat{e}$ by other points with strength $E_s \geq t_l$ in the direction of the edge at $\hat{e}$.

Once this five step process is complete, a binary image is generated where a ‘1’ represents an edge cell.

This process is necessary because the occupancy grid mapping shows boundaries of obstacles as thicker than they are in reality due to the noise in sensor measurements. Using edge detection, it is possible to extract the exact obstacle boundaries to be used in the other blocks so that the map can be processed more accurately and efficiently.

3.3 Edge Smoothing

The next block is **Edge Smoothing**. Here, the Canny edges are used to estimate the most probable locations of the obstacles. The objective is to remove redundant edges caused by the noisy laser sensor measurements and select the most likely locations of the obstacles.

The error of laser measurements follows a Gaussian distribution. As an example, Fig. 3-a depicts a 2D Gaussian distribution along a curve where the distribution at each point is given by:

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2} - \frac{(y-\mu_y)^2}{2\sigma_y^2}}, \tag{13}$$

where σ_x and σ_y are variance values and μ_x and μ_y are mean values for two uncorrelated variables, x and y.

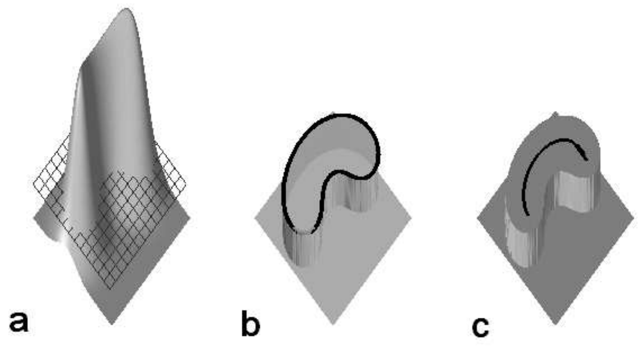

Figure 3: **a)** A 2D Gaussian distribution with no truncation. The meshed plane is the truncation surface. **b)** Any point below the surface is set to zero to produce a truncated Gaussian distribution. **c)** Middle point of edges.

Through the edge detection operation, the Gaussian distribution from Fig. 3-a is truncated based on the Canny threshold t_h and t_l to produce the edges shown in Fig. 3-b. However, the edge detection algorithm has produced two edges where only one obstacle exists in reality. These two edges are reduced to one, which resides at the peak of the Gaussian distribution as shown in Fig. 3-c.

In the implementation, it is therefore reasonable to assume that pairs of nearby edges actually represent only a single edge, which can be assumed to be at a location equidistant from the two original (redundant) edges. The location of the new edge is found by performing averaging in the horizontal and vertical directions as described in detail in Algorithm 1.

More specifically, suppose that we have two adjacent edges $\mathcal{C}_1$ and $\mathcal{C}_2$ that are output from the Canny edge detector. Assuming that the two edges are close enough that they actually only represent one true obstacle, the most probable position of the

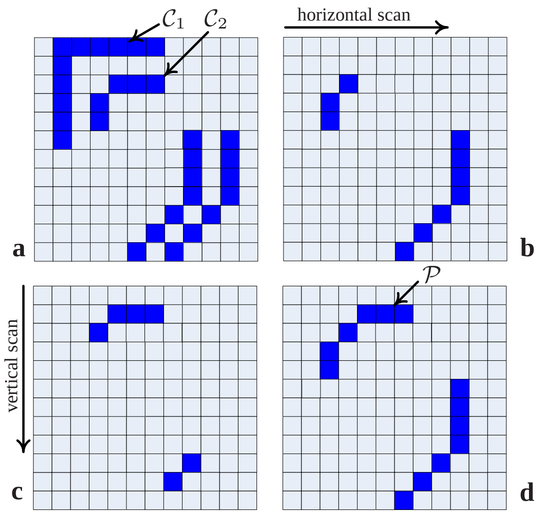

Figure 4: Vertical and horizontal scans for a simple edge map ($d_t = 6$). **a)** Original map: two adjacent edges, $\mathcal{C}_1$ and $\mathcal{C}_2$ are shown in the map. **b)** Output after the horizontal scan. **c)** Output after the vertical scan. **d)** Output after both scans. Final $\mathcal{P}$ curve for $\mathcal{C}_1$ and $\mathcal{C}_2$, mentioned in (14), is the output of the process.

obstacle, $\mathcal{P}$, is given by

$$\mathcal{P} = \mathcal{C}_1 \oplus \mathcal{C}_2, \tag{14}$$

where the operator $\oplus$ performs middle curve extraction on two dimensional curves of $\mathcal{C}_1$ and $\mathcal{C}_2$ by doing an averaging in the horizontal and vertical directions, and outputs the set of points $\mathcal{P}$ that represent the best approximation of the actual location of the obstacle.

It is difficult to generate closed mathematical representations of $\mathcal{C}_1$ and $\mathcal{C}_2$, so the operator $\oplus$ can be defined as operating over the detected edges by scanning in the vertical and horizontal directions. Algorithm 1 explains this process for a vertical scan. The input to the algorithm is the edge map and the output is a smoothed edge map. First the output matrix is initialized to a zero matrix (line 1). In a vertical scan, (lines 2 to 17), all columns are processed for each row. Occupied cells in each row are identified in line 5 and stored in a temporary variable (line 7). Once two occupied cells are

Algorithm 1 Scanning an edge map in a vertical direction.

```
Input: Edge map: $E_{R\times C}$,
Output: Smoothed Edge map $S_{R\times C}$,
1: $S(r,c) \leftarrow 0, \forall(r_{1:R}, c_{1:C})$
2: for $r = 1 \rightarrow R$ do
3:     $n \leftarrow 0, cell \leftarrow \emptyset$
4:     for $c = 1 \rightarrow C$ do
5:         if $E(r,c) = 1$ then
6:             $n \leftarrow n + 1$
7:             $cell(n) \leftarrow c$
8:             if $n = 2$ then
9:                 if $1 < cell(n) - cell(n-1) < d_t$ then
10:                    $c_s \leftarrow round((cell(n) + cell(n-1))/2)$
11:                    $S(r, c_s) = 1$
12:                end if
13:                $n \leftarrow 0, cell \leftarrow \emptyset$
14:            end if
15:        end if
16:    end for
17: end for
```

identified (line 8), their distances are processed. If the distance between two occupied cells is larger than one and less than some specified threshold d_t, (line 9) then the middle point of the two occupied cells is considered to be the most probable occupied cell (line 10) and this cell is marked as an occupied cell in the output map (line 11). This process continues for all cells in each row. A similar process is performed in the horizontal direction. Fig. 4 shows a simple example of the scanning process ($d_t = 6$). The major benefit of smoothing edges is that the complexity and scale of the resulting maps are reduced without any significant loss of data.

One major advantage of this approach is that the location of resulting edges should be invariant to the quality of the sensor used, since noise is being filtered out. This is important for map merging in the subsequent steps since if robots have variable sensing capabilities, they should still be able to merge their maps.

3.4 Segmentation

The next block is the **Segmentation** block. The objective of this block is try to select unique areas of each map so that they can be compared in the hope of finding parts of the maps that are shared. These shared features are used to find the relative transformation matrix between the two robots. The Segmentation block looks for segments of the map which have unique geometric properties, similar to the way the human brain functions [17]. As a human being, our brain tries to find shared parts in different images and then identifies the transformation based on those shared parts. Specifically, parts of the map that contain curves or angles are selected as good candidate segments. It is also noted that segments should be made from the occupied cells. Because a normal occupancy grid map will have significantly less occupied cells than free or unknown cells.

The segment selection process begins with choosing a random start point in the map, which can be any point that represents an occupied cell. A fixed number of points, n, along the edge are then identified and represented as:

$$P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, i = 1..n, \tag{15}$$

where P_i is the i^{th} point of the segment and x_i and y_i denote the discrete points and P_0 is the first point of the segment which is a reference for other points.

The segment is then processed to determine if it contains unique geometrical characteristics by generating and analyzing two other vectors: the Euclidian distance vector, D and the differential angle vector, Δ:

$$\boldsymbol{D} = \begin{bmatrix} d_1 & d_2 & \dots & d_n \end{bmatrix},$$

$$\boldsymbol{\Delta} = \begin{bmatrix} \delta_1 & \delta_2 & \dots & \delta_{n-1} \end{bmatrix}, \tag{16}$$

where

$$d_i = ||P_i - P_0||,$$

$$\delta_i = \angle(P_i - P_0) - \angle(P_{i-1} - P_0). \tag{17}$$

Fig. 5-a illustrates d_i and δ_i for three points. The vector $\boldsymbol{D}$ of the segment is invariant with respect to any rotation or translation of the segment. The vector Δ will differentiate between two segments which are mirror images.

The two characteristic vectors are a unique representation, $\mathbf{\Lambda}$, of the original segment P as given by:

$$\mathbf{\Lambda} = (\boldsymbol{D}, \mathbf{\Delta}). \tag{18}$$

Theorem 1. *From $\mathbf{\Lambda}$ and the starting location of the segment, P_0, it is possible to reconstruct the entire segment P uniquely.*

Proof. By induction:

Base case

P_0 must be known.

Induction step

Assume that k points in the segment are known. From d_{k+1}, it is known that the point P_{k+1} lies on a circular locus of radius d_{k+1} centered at P_0 (see Fig. 5-b). Consider that points P_0 and P_k are already located. Let $\phi_k = \angle(P_k - P_0)$, which is the known angle of point k given that its exact location is already known. Let $\phi_{k+1} = \angle(P_{k+1} - P_0)$, which is unknown. From (17), $\delta_{k+1} = \phi_{k+1} - \phi_k$. Rearranging gives $\phi_{k+1} = \delta_{k+1} + \phi_k$. Since both terms on the right hand side are known, the angle ϕ_{k+1} can be calculated. From the angle that P_{k+1} makes with P_0, the point on the circular locus can be uniquely identified. $\square$

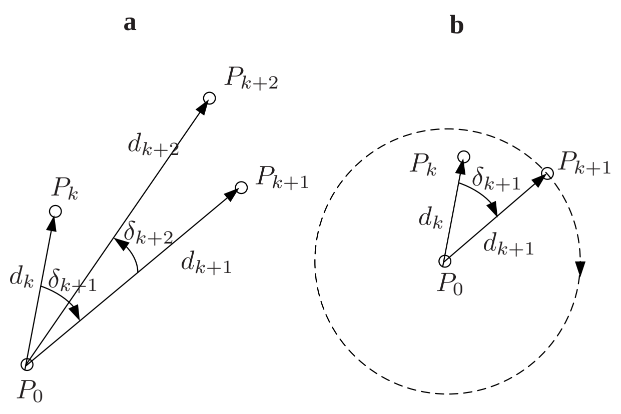

Figure 5: A simple example of the segmentation. **a) differential angles. b) reconstructing a segment.**

3.5 Segment Verification

To identify whether a segment is a good candidate for map fusion, a histogram method is used. Two histograms are constructed to verify a segment. The first one is the distance histogram, $hist_d$ which is generated from the distance vector, $\boldsymbol{D}$, and the second is the arc histogram, $hist_a$, which is generated from the arc vector which is defined as:

$$\boldsymbol{\Omega} = \begin{bmatrix} \omega_1 & \omega_2 & \dots & \omega_{n-1} \end{bmatrix}. \tag{19}$$

Each arc is defined as

$$\omega_i = d_{i+1} \sum_{j=1}^{i} \delta_j, \tag{20}$$

where $i = 1, 2, \dots, n-1$.

The histograms are generated using a tunable number of bins, where the values in the bins correspond to the number of elements of the vector that fall between the boundaries specified for that bin. The number of bins should be chosen to sufficiently represent the information in the segments. Algorithm 2 explains this process for a $1 \times N$ vector, $v_{1\times N}$ with a $1 \times M$ bin vector specified by $bin_{1\times M}$. The output of the algorithm is the histogram depicted by a vector $hist_{1\times M}$. The number of bins chosen is important in order to get acceptable result. In this research, it is determined experimentally over a set of trials.

Algorithm 2 Generating a histogram for a vector.

```
Input: $v_{1\times N} = \{v_n\}_{n=1}^{N}, bin_{1\times M} = \{b_m\}_{m=1}^{M},$
Output: $hist_{1\times M} = \{h_m\}_{m=1}^{M},$
1: $h_m \leftarrow 0, \forall h_m, m = 1, ...M$
2: for $n = 1 \rightarrow N$ do
3:     for $m = 1 \rightarrow M$ do
4:         if $(v_n \geq b_m)\&(v_n < b_{m+1})$ then
5:             $h_m \leftarrow h_m + 1$
6:         end if
7:     end for
8: end for
```

The distance histogram, $hist_d$, and arc histogram, $hist_a$ are represented as:

$$hist_d = \begin{bmatrix} hd_1 & hd_2 & \dots & hd_{nd} \end{bmatrix},$$

$$hist_a = \begin{bmatrix} ha_1 & ha_2 & \dots & ha_{na} \end{bmatrix}, \tag{21}$$

where nd is the number of the bins and $hd_j, j = 1, 2, \dots, nd$ is the value corresponding to each bin for $hist_d$, and na is the number of the bins and $ha_k, k = 1, 2, \dots, na$ is the value corresponding to each bin for $hist_a$. Fig. 6-b and Fig. 6-c show distance and arc histograms of the segment shown in Fig. 6-a.

If the segment is a wall or flat surface, then the distance histogram will have a flat shape, and the arc histogram will have almost all bins empty except for one. If there is an angle in the segment, then the distance histogram will not be flat and more than one bin of the arc histogram will be non-empty.

To verify the acceptability of a segment, the following conditions should be met:

1) $hist_d(j) \neq 0, \forall j = 1 \dots nd$

2) The distance histogram should not be smooth:

$$\max(hist_d) - \min(hist_d) \geq h_d, \tag{22}$$

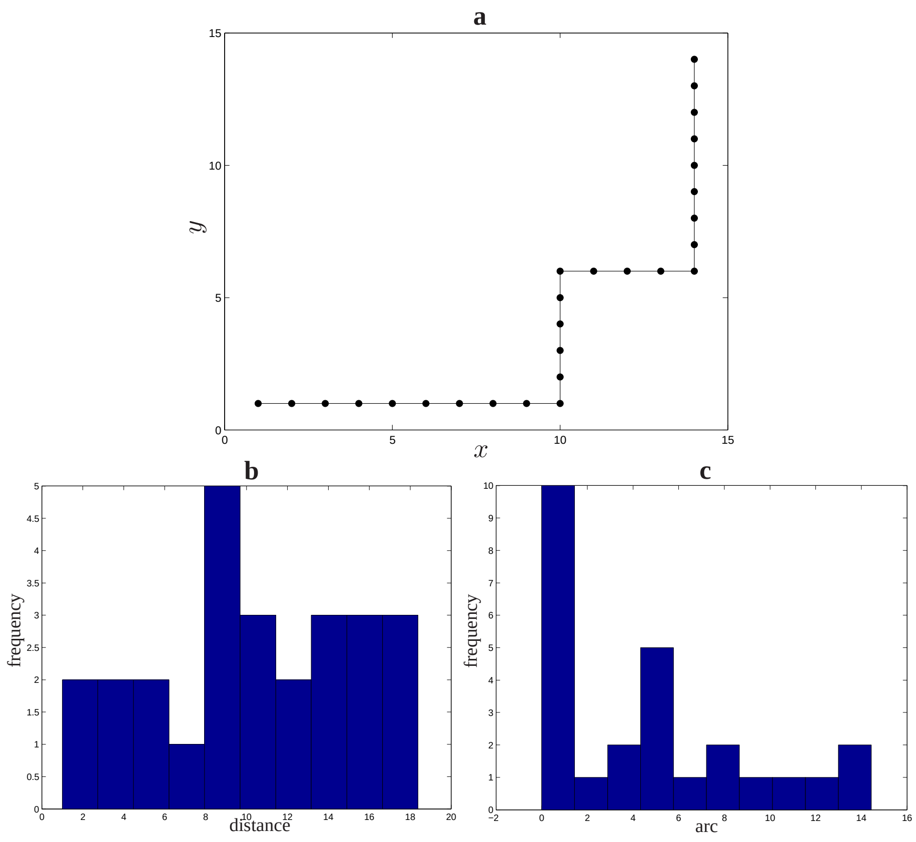

Figure 6: An example of distance and arc histograms. **a)** Points represent a segment. **b)** Distance histogram of the segment **c)** Arc histogram of the segment.

where h_d is a predefined threshold value.

3) The segment should not be discontinuous. A discontinuity will cause a large value in the arc vector, so the condition:

$$\max(|\omega_i|) \leq \omega_c, \tag{23}$$

where ω_c is a predefined threshold value that will reject segments with discontinuities. For instance, by choosing $h_d = 3$, and $\omega_c = 11$, histograms shown in Fig. 6-b and Fig. 6-c will verify the segment shown in Fig. 6-a. Selection of these parameters are discussed in Section 4.4.

This novel histogram method is fast and is successful at identifying segments with unique geometric properties.

It is important to note that the size of the occupancy grid map cells is fixed across all robots in the team. The result is that matching segments will be the same size across all maps.

3.6 Cross Correlation

In the **Cross Correlation** block, the distance and arc histograms of the two selected segments from the two maps are compared using a cross correlation technique. The histograms are normalized and then a 2D cross correlation operation is applied. If the distance and arc histograms are similar, then the result of each cross correlation will be close to one and the segments are deemed a match. Otherwise, another segment from map_2 is selected and the process continues. If, for the selected segment from map_1, no correspondence from map_2 can be found, then another segment from map_1 should be selected. If no shared segment can be found after a certain number of trials, then the two maps cannot be fused. To increase the confidence level of the result of the histogram matching, a correlation of the distance and arc vectors is also applied.

Given two vectors, $u_{1\times N}$ and $v_{1\times N}$ the following relation is used to calculate the correlation coefficient as a measure of similarity.

$$r = \frac{\sum_{n=1}^{N}(u_n - \bar{u})(v_n - \bar{v})}{\sqrt{(\sum_{n=1}^{N}(u_n - \bar{u})^2)(\sum_{n=1}^{N}(v_n - \bar{v})^2)}}, \tag{24}$$

where $\bar{u}$ and $\bar{v}$ are mean values of vectors u and v respectively.

Fig. 7-a and Fig. 7-b show two segments, chosen for cross correlation. The matching score of the segments, based on equation (24), is 0.95 (the maximum matching score is 1), which means the segments are verified and can be used to calculate the relative transformation. Note that if there are multiple matching segments, for each matching pair, a transformation is calculated based the rest of the algorithm. At the end, only the transformation of the matching segment which results in the best verification index, introduced in Section 3.9, is accepted.

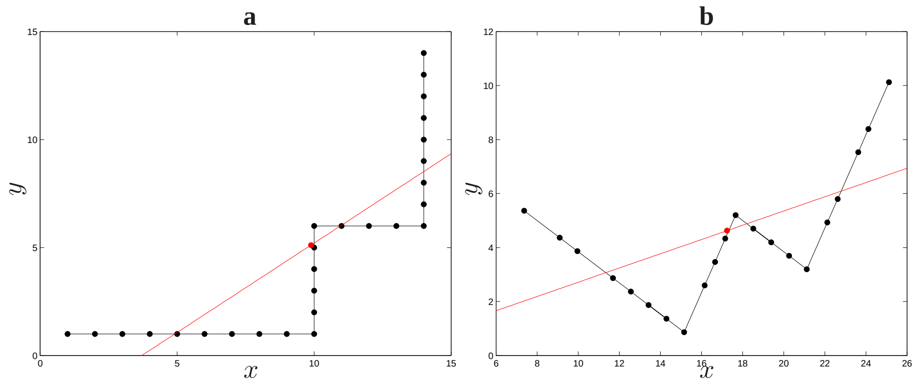

Figure 7: Two matching segments **a)** The first segment. **b)** The second segment. In both figures, the centroids of the segments are indicated by red dots and are used to calculate the relative translation. The difference of the slope of the fitted lines, shown in red, is used to calculate the relative rotation.

3.7 Pose Extraction

After finding matching segments from both maps, the **Pose Extraction** block is executed. First, the approximate relative rotation is determined, and then the rotation is used to find the approximate translation. To determine the rotation, a line is fitted for each segment using minimum least-squares error. The difference in the slopes of the two fitted lines gives the approximate rotation angle between the two maps. If the slope of the fitted line for the selected segment of map_1 is a_1 and of map_2 is a_2, then the approximate rotation angle, α_{app} is:

$$\alpha_{app} = \text{arctan2}(a_1) - \text{arctan2}(a_2) \tag{25}$$

where $\operatorname{arctan2}$ is the arc tangent function that bounds the output to the interval $(-\pi, \pi]$. Fitted lines to segments in Fig. 7-a and Fig. 7-b are shown by red lines. These lines are used to calculate the relative rotation using equation (25).

Once the approximate rotation has been found, the approximate translation can be determined. First, the approximate rotation is applied to align both segments. Next, the approximate translation is computed to be the difference in the centroids of the two segments as given by:

$$T_{app} = \begin{bmatrix} x_{app} \\ y_{app} \end{bmatrix} = \sum_{k=1}^{n_1} \frac{R_{\alpha_{app}} P_k^{seg1}}{n_1} - \sum_{k=1}^{n_2} \frac{P_k^{seg2}}{n_2}, \tag{26}$$

where $R_{\alpha_{app}}$ is the rotation matrix based on α_{app}, P^{seg1} and P^{seg2} are the sets of points in segments 1 and 2 respectively. n_1 and n_2 are the cardinalities of P^{seg1} and P^{seg2} respectively.

The centroids of the segments in Fig. 7-a and Fig. 7-b are highlighted by red points. The centroids are used to calculate the relative translation using equation (26).

3.8 Pose Tuning

After finding the approximate rotation and translations, these elements need to be tuned. The tuned pose will be extracted in two steps using the *Radon Transform* and a *Similarity Index*. First, to find the tuned rotation a Radon transform is used [27]. The Radon transform is the projection of the image intensity along a radial line oriented at a specific angle. The Radon image is generated by computing a Radon transform and varying the Radon angle, θ, of the radial line onto which the original image is being projected. For a given map, $m(x, y)$, the Radon transform along the radial line of angle θ is defined as:

$$r_\theta(x') = \int_{-\infty}^{\infty} m(x' \cos\theta - y' \sin\theta, x' \sin\theta + y' \cos\theta) dy', \tag{27}$$

where

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{28}$$

The Radon image generated by computing a Radon transform is a collection of all possible projections for a larger set of angles. In a Radon image, the horizontal axis represents all possible angles and the vertical axis is the Radon transform for individual angles.

One of the characteristics of the Radon image is that the peak points occur when the Radon angle, θ, aligns with straight line segments that occur in the image. The approximate rotation is used to ensure that the correct peak is selected from the Radon transform.

As a result, it is possible to resolve the exact rotation by looking for peaks in the Radon images of both maps that are close to the approximate angle already computed.

$$\alpha_{ext} = \alpha_{app} + \delta_{tuning}, \tag{29}$$

$$\delta_{tuning} = \arg\min_{\theta}(\mathcal{R}_{\theta=0:180}(m_1)) - \arg\min_{\theta}(\mathcal{R}_{\theta=0:180}(\mathcal{T}_{x_{app},y_{app},\alpha_{app}}(m_2))), \tag{30}$$

where α_{ext} is the exact or tuned rotation angle and $\mathcal{R}_{\theta=0:180}(map)$ is the Radon image of the input map over the specified domain of angles (θ). $\mathcal{T}_{x,y,\theta}(map)$ means that the input map is translated according to the values of x, y and rotated by θ and $m_1 = map_1$ and $m_2 = map_2$.

After tuning the rotation, the translation is tuned. Here two methods are proposed for translation tuning. The first method uses a **similarity index**. A performance index is used which has been introduced in [4]. This index measures the similarity of two maps over some desired region. When there is more overlap between two maps, there will be an extremum in the index. This index is composed of two components:

$$\begin{aligned} agr(map_1, map_2) &= \sharp\{p = (x, y) | map_1(p) = map_2(p)\}, \\ dis(map_1, map_2) &= \sharp\{p = (x, y) | map_1(p) \neq map_2(p)\}, \end{aligned} \tag{31}$$

where the operator $\sharp$ over a given set returns the cardinality of the set. The similarity index is defined as:

$$J_{sim}(map_1, map_2) = dis(map_1, map_2) - agr(map_1, map_2). \tag{32}$$

The function $agr(\cdot)$, the agreement index, is the number of known cells with equal status in both maps (either both occupied or both free). The function $dis(\cdot)$, the disagreement index, is the number of cells which are known in at least one map and have unequal status. A smaller disagreement index and larger agreement index will result in a larger similarity index, J_{sim}. The maximum absolute value represents the best translational match between the two maps. To find the maximal value, an exhaustive search in the neighborhood of the approximate translation is done. It is challenging to use guided search algorithms like Genetic Algorithms (GA) or Particle Swarm Optimization (PSO) because in many cases the best solution may occur very close to very poor solutions. The size of the search space is determined by experience based on the performance of the approximate translation method. In addition, if the total allowable time for searching can be determined, the size of the search space can be specified such that the search is completed within the allotted time. Usually a search space with a radius of ten cells around the current approximate solution is acceptable considering the required time constraints and accuracy of the previous approximate result. The best candidate translation vector is selected by determining the one with the best similarity index:

$$T_{ext} = \begin{bmatrix} x_{ext} \\ y_{ext} \end{bmatrix} = \underset{\mathcal{S}}{\operatorname{argmax}}(|J_{sim}(m_1, \mathcal{T}_{x,y,\alpha_{ext}}(m_2))|), \tag{33}$$

$$\mathcal{S} = \{(x, y) \subset \mathbb{N}^2 | x_{app} - \delta_x < x < x_{app} + \delta_x,$$
$$y_{app} - \delta_y < y < y_{app} + \delta_y\} \quad (34)$$

where T_{ext} is the tuned translation vector and $\mathcal{S}$ is the search space, defined as a rectangle centered at (x_{app}, y_{app}) with dimensions $2\delta_x \times 2\delta_y$.

The second method of the translation tuning proposed in here is based on the image entropy. Image entropy for an input map is defined as:

$$\mathcal{H}(map) = -\sum p \log_2(p), \quad (35)$$

where p is the normalized histogram of the map. Entropy is a statistical measure of the randomness associated with the map and is usually applied to characterize the texture.

An accurate translation corresponds to the minimum entropy, therefore the following relation gives the best translation vector:

$$T_{ext} = \begin{bmatrix} x_{ext} \\ y_{ext} \end{bmatrix} = \underset{\mathcal{S}}{\operatorname{argmin}} \{\mathcal{H}(\mathcal{F}(m_1, \mathcal{T}_{x_x, y_y, \alpha_{ext}}(m_2))\}$$

$\mathcal{F}(m_1, m_2)$ means both maps are fused using (9) and are in the same coordinate frame according to the coordinates of the first input, m_1. $\mathcal{H}(\cdot)$ is the entropy of the image as defined in equation (35). $\mathcal{S}$ is the same as equation (34) and T_{ext} is translation vector to be verified.

3.9 Pose Verification

After finding the best candidate for the transformation matrix, a verification is performed in the **Pose Verification** block. In case of false matching, a verification is

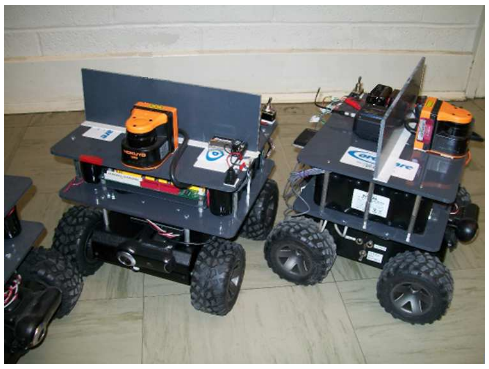

Figure 8: Robots equipped with the laser rangefinder, MEMS IMU and wheel encoders (not visible)

required to ensure the results are accurate. The verification process acts as a check on whether the previous blocks operated correctly. If the percentage of the verification is lower than a threshold, the results are rejected and the process can start over with other segments or at a later time when the maps have more overlaps. The verification method, called similarity verification, uses the ratio of $agr(\cdot)$ and $dis(\cdot)$ as given by:

$$V(m_1, m_2') = \frac{agr(m_1, m_2') \times 100\%}{agr(m_1, m_2') + dis(m_1, m'2)},$$

$$m_2' = \mathcal{T}_{x_{ext}, y_{ext}, \alpha_{ext}}(m_2) \tag{36}$$

where m_1 and m_2 are two input maps and $V(m_1, m_2')$ is the verification index. m_2' is translated according to x_{ext} and y_{ext} and rotated based on α_{ext}. If $V(m_1, m_2')$ is more than a threshold, then the proposed transformation can be accepted. The maximum verification percentage occurs when the maps are perfectly aligned.

4 Experimental Results for Validation

In this section, first hardware and software setups are explained, then the results of two real world experiments are presented. In the end, comparison results are explained.

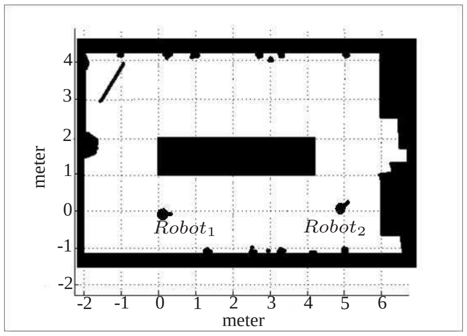

Figure 9: A simple test environment with two robots for experiment 1.

4.1 Experimental Setups

In this research, multiple differential-wheeled robots are used. The robots are built by CoroWare, Inc. and each is equipped with three types of sensors: two High Speed Phidget encoders on the wheels, a UBG-05LN laser rangefinder from Hokuyo, and a 3DM-GX1 Microstrain IMU, as shown in Fig. 8. The laser rangefinder sensor can measure up to 4500 millimeters range over a sector of 184.32^o. The angular resolution is 0.36^o resulting in 513 points for each scan. The IMU is based on Microelectromechanical Systems (MEMS) and contains three types of integrated sensors: accelerometer, gyroscope and magnetometer. Each of these gives measurements along the x, y and z axes of the body frame, for a total of nine measurements. Data from these sensors are fused with a cascaded EKF. Specifically, data from the laser rangefinder and IMU are filtered sequentially to provide an estimate of the pose of the robot.

4.2 Experiment 1: Simple Test Environment with Two Robots

Fig. 9 shows the approximate floor plan and dimensions of an indoor environment used for experimentation. This room has been surveyed by two robots from different initial positions which are assumed unknown.

Fig. 10-a shows the local map provided by $robot_1$, and Fig. 11-a shows the local map provided by $robot_2$. In both figures, gray cells are unknown, white cells are free,

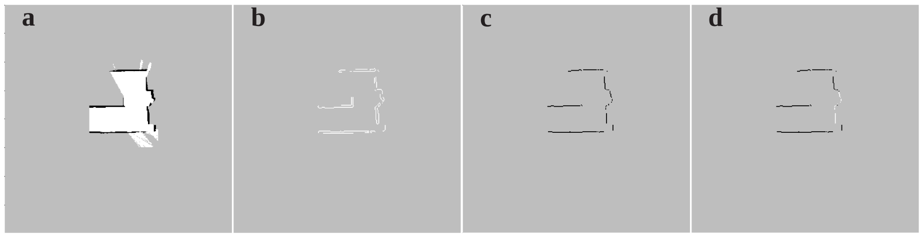

Figure 10: **a)** $\mathbf{m}_1$, occupancy grid map of the test environment provided by the first robot. **b)** Canny edges **c)** Smoothed edges **d)** Matching segment in white color

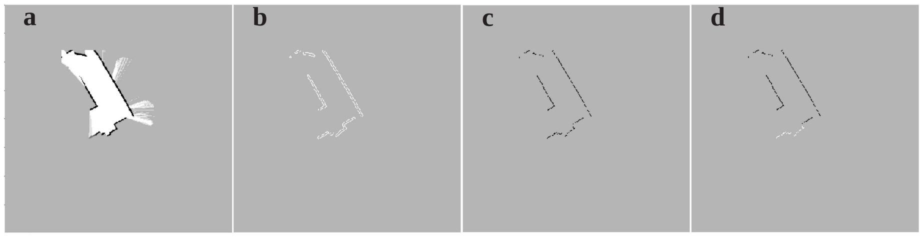

Figure 11: **a)** $\mathbf{m}_2$, occupancy grid map of the test environment provided by the second robot **b)** Canny edges **c)** Smoothed edges **d**) Matching segment in white color

and black cells are obstacles. The size of all maps is 400×400 cells with identical resolution.

Fig. 10-b and Fig. 11-b show the effects of the *Edge Detection* block on the two maps. These figures show boundaries of occupied cells which will be used in the next block.

Fig. 10-c and Fig. 11-c show the results of the *Edge Smoothing* block for map_1 and map_2, respectively. Extracted edges are smoothed to remove redundant edges caused by sensor noise.

Fig. 10-d and Fig. 11-d show two matching segments from map_1 and map_2 in white which are the results of the *Segmentation*, *Segment Verification* and *Cross Correlation* blocks.

The criterion to select these segments for matching is implemented in the *Segment Verification* and *Cross Correlation* blocks. In the first block, arc and distance histograms are used to verify a segment. As explained, a segment with a flat distance

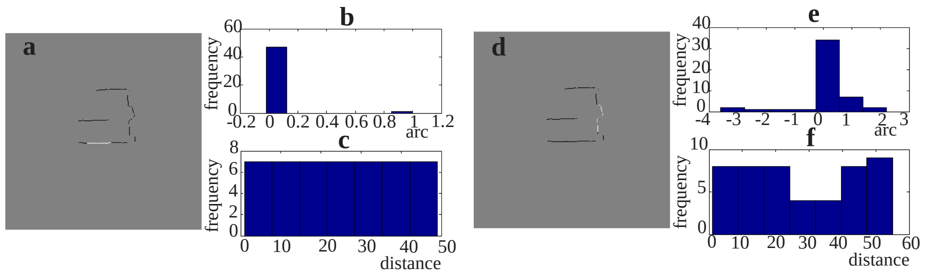

Figure 12: **a)** A straight wall segment **b)** Arc histogram of the straight segment **c)** Distance histogram of the straight segment, **d)** A featured wall segment **e)** Arc histogram of the featured segment **f)** Distance histogram of the featured segment

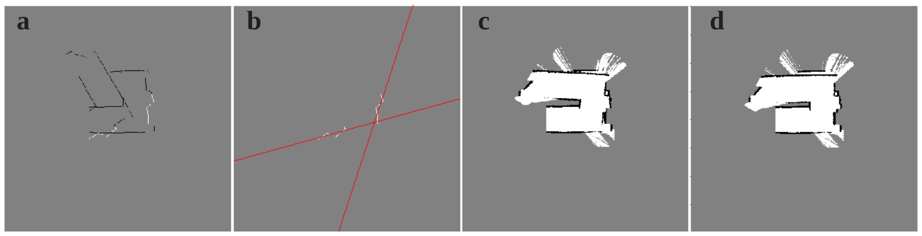

Figure 13: **a)** Both maps in the same coordinates with marked segments, **b)** Fitted lines on segments to extract relative orientation, **c)**, Both maps in the coordinates of $\mathbf{m}_1$ after approximate transformation, **d)** Rotation improvement after Radon transformation

histogram or a sharp arc histogram is not a good segment for map merging. Fig. 12-a shows a flat segment with its distance and arc histograms in Fig. 12-b and Fig. 12-c, respectively. Fig. 12-d shows a non-flat segment. Its distance and arc histograms are depicted in Fig. 12-e and Fig. 12-f, respectively. The flat segment is not a good candidate for map merging while the second segment is. If segments have acceptable histograms, they are matched by the cross correlation operation and if the similarity is high enough, they are used for map merging.

Fig. 13-a shows both maps on the coordinates of map_1 with identified common segments. Fig. 13-b shows the fitted lines of the two similar segments. The angle between these two lines is an approximation of the relative orientation of the two maps. After finding the approximate rotation, equation (26) is used to find the approximate translation. These operations are performed in the *Approximate Pose Extraction* block.

Fig. 13-c shows both maps in the coordinates of map_1 after the approximate transformation. It is clear from the figure that this transformation is not accurate and needs further modification.

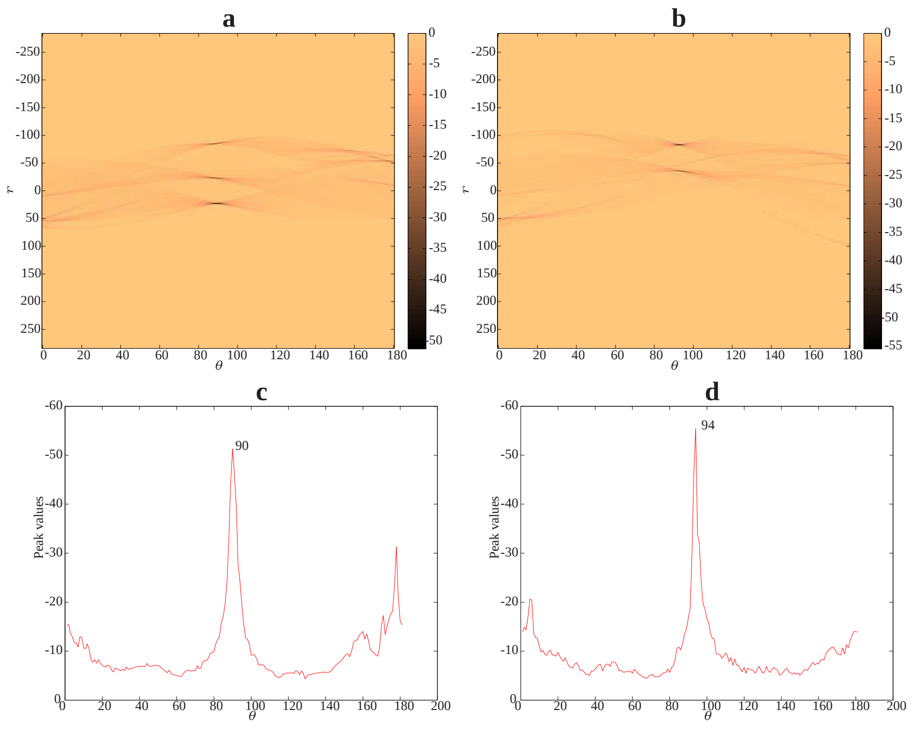

Figure 14: **a)** Radon image of $\mathbf{m}_1$ over 180 degrees **b)** peak point of the Radon image

The next step is tuning the rotation angle and the translation which are performed in the *Pose Tuning* block. First the orientation is tuned by the Radon transform. Fig. 14-a and Fig. 14-b show the Radon images for map_1 and map_2 respectively after applying the approximate transformation. As Fig. 14-c and Fig. 14-d show, the peak point of the Radon image for map_1 happens to be at 90 degrees while for the approximately transformed map_2 it is 94 degrees. The difference of 4 degrees is the tuning angle that should be added to the original approximate angle as determined by equation (29).

Fig. 13-d shows both maps in the coordinates of map_1 after applying the proposed Radon tuning method. Clearly, this figure shows the effective improvement over the relative rotation angle shown in Fig 13-c.

The next step is tuning the translation. Fig. 15 shows a 3D representation of the similarity indices over a set of translation vectors along the vertical and horizontal axes following a raster scan pattern. As indicated, the best similarity index occurs at the best translation, which is shown by a circle.

As an alternative method for the translation tuning, Fig. 16 shows the entropy of both maps when they are in the coordinates of map_1. Once again it is noted that the minimum entropy corresponds to the transformation matrix that was found.

Finally, Fig. 17 shows the final overlapped maps with exact rotation and translation elements.

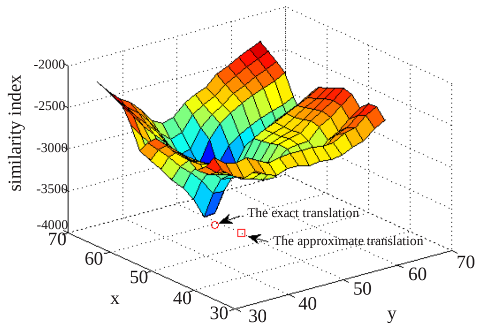

Figure 15: 3D representation of the similarity index over the search space. The best similarity index occurs at the best transformation.

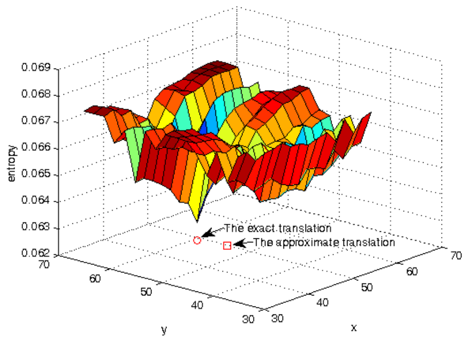

Figure 16: 3D representation of the entropy of the fused maps over the search space. The best translation occurs at the minimum entropy.

To verify the results, the proposed method in the *Pose Verification* block is used. The verification index using the similarity index introduced in equation (36) is 95% which is the peak of all verification indices over the search space. This is shown in Fig. 18.

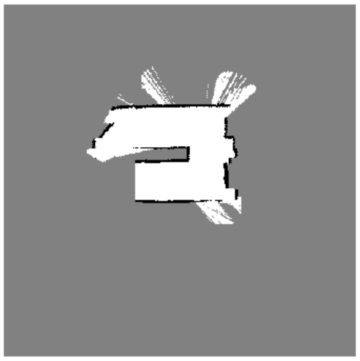
Figure 17: Final alignment of both maps in the coordinates of map_1

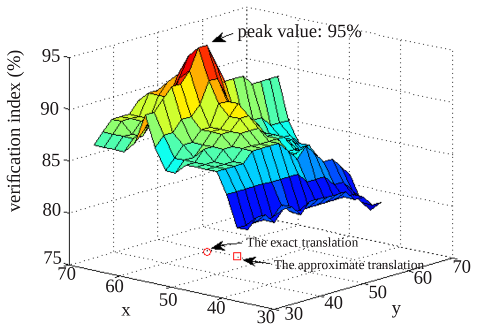

Figure 18: 3D representation of the verification index over the search space for the fused maps. The maximum value occurs at the best translation.

4.3 Experiment 2: More Complex Environment with Three Robots

To demonstrate the effectiveness of the proposed methods, an experiment is performed using three robots in an environment with the approximate size of 17×10 meters. Fig. 19 shows the approximate floor plan of the test environment.

Each robot generates a local occupancy grid map of the environment. At certain specified time intervals, the robots share their local maps with each other. Fig. 20 shows the three local maps generated by the three robots. Note that the left part of the test environment was out of the range of the laser rangefinders for all three robots. The three maps. map_1, map_2 and map_3, are provided by $robot_1$, $robot_2$ and $robot_3$ respectively. The proposed map fusion algorithm is assumed to be done on $robot_1$ with map_2 and map_3 being integrated into map_1.

Fig. 21-a shows the result of map fusion of map_2 into map_1 using the approximate translation and tuned rotation: $\boldsymbol{T}_{x,y,\theta} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T = \begin{bmatrix} 43 & -13 & 19.8 \end{bmatrix}^T$.

After applying the proposed tuning method for translation, the tuned transformation becomes $\boldsymbol{T}_{x,y,\theta} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T = \begin{bmatrix} 40 & -16 & 19.8 \end{bmatrix}^T$. Fig. 21-b shows the exact alignment for map_2 into map_1. The verification index after final alignment is 97% and the results of the similarity index and image entropy are consistent.

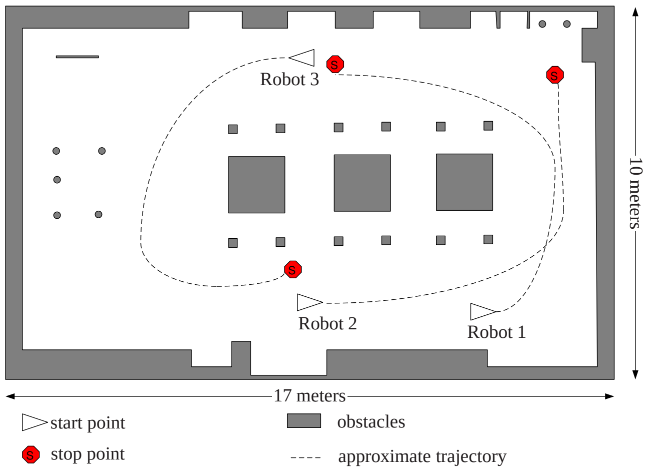

Figure 19: Floor plan of the real world test environment. Start and stop points of the robots are marked with triangles and stop signs, respectively. The trajectory of each robot is depicted with dashed lines and obstacles are shown in dark color. All markings are approximate.

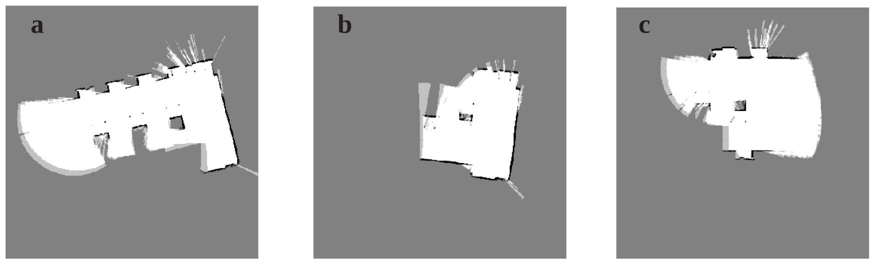

Figure 20: Three local maps provided by three robots based on the approximate dimensions and trajectories of Fig. 19. **a)** $\mathbf{m}_1$ provided by $robot_1$, **b)** $\mathbf{m}_2$ provided by $robot_2$, **c)** $\mathbf{m}_3$ provided by $robot_3$

The next step is to fuse map_1 with map_3. Similarly Fig. 22-a shows results of map fusion with the approximate translation and tuned rotation. Fig. 22-b shows the result of the tuned alignment. Approximate and exact translations for the fusion of map_1 and map_3 are $\begin{bmatrix} -74 & 23 & 192.8 \end{bmatrix}^T$ and $\begin{bmatrix} -67 & 32 & 192.8 \end{bmatrix}^T$, respectively. After the final alignment the verification index is 98%. The proposed entropy method is verified by the results.

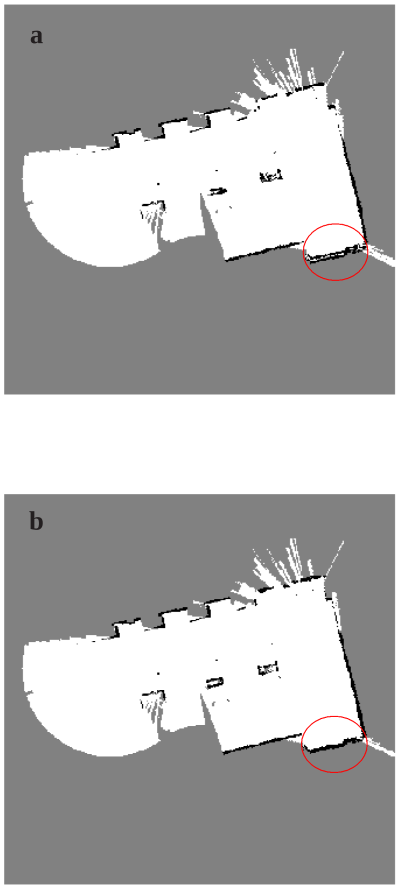

Figure 21: Map fusion for $\mathbf{m}_1$ and $\mathbf{m}_2$ based on the local maps presented in Fig. 20-a, b **a)** alignment of two maps after approximated translation and tuned rotation, **b)** alignment of two maps after exact transformation. Differences between the approximate and tuned maps are highlighted.

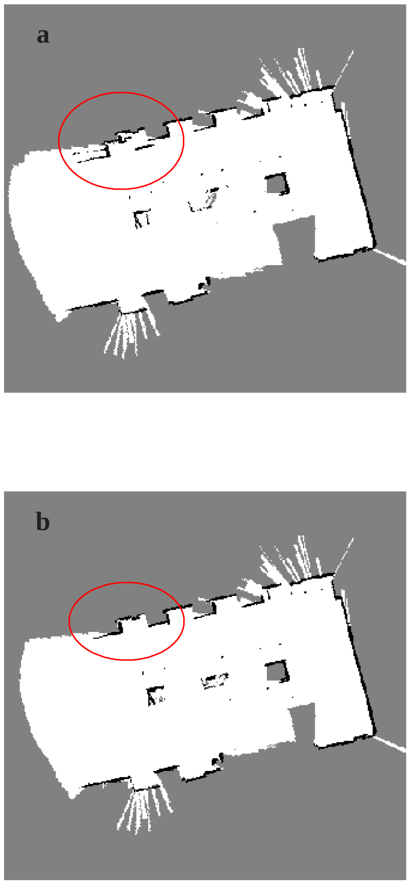

Figure 22: Map fusion for $\mathbf{m}_1$ and $\mathbf{m}_3$ based on the local maps presented in Fig. 20-a, c **a)** alignment of two maps after approximated translation and tuned rotation, **b)** alignment of two maps after exact transformation. Differences between the approximate and tuned maps are highlighted.

From the Fig. 21 and Fig. 22, it is clear the "approximate" maps are quite accurate. This is generally the case. However, if the approximate rotation or translations are highly inaccurate, it can result in higher computation time in the tuning step due to the increased search space.

Finally, Fig. 23 shows all three local maps fused together and shown in the local coordinates of map_1.

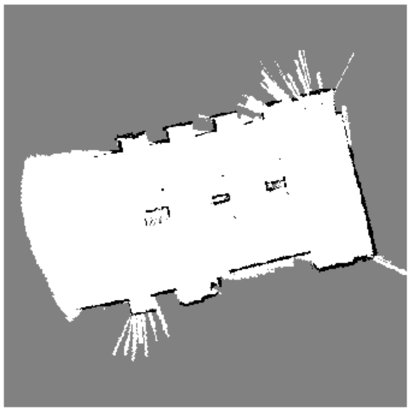

Figure 23: Map fusion for $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_3$ based on the local maps presented in Fig. 20-a, b, c; $\mathbf{m}_2$ and $\mathbf{m}_3$ are transformed to the coordinates of $\mathbf{m}_1$ after determining the tuned transformation matrices

It is clear from the results that the three robot team is able to map the environment more quickly than one robot alone. Each robot in the team generates a map of some part of the workspace and then, if there is some overlap between the maps, all local maps can be fused into a global map. It is important to note that the robots are not required to meet at any point to determine their relative positions, this information is generated from the maps themselves.

4.4 Discussion: Parameter Selection

Throughout the algorithm, many parameters are introduced which need to be tuned for good performance. Table 1 lists these parameters. These parameters have been tuned by trial and error; however, it is possible to use non-derivative optimization methods which is beyond the scope of the paper. Repeated experiments show that changing these parameters in a small interval does not affect the quality of the output significantly. The pairs, in the “Value” column of Table 1, correspond to minimum and maximum values of the parameter to generate acceptable results. The value of t_h, mentioned in the first row of Table 1, is relative to the maximum of the gradient magnitude of the map, defined in equation (11) and calculated online during the edge detection process. In general, these parameters are dependent on the map resolution and sensor characteristics and need to be tuned properly.

Table 1: List of important parameters with their acceptable minimum and maximum values. The pairs, stated for most parameters, correspond to minimum and maximum values of the parameter which generate acceptable results. For instance, (90, 97) for the Pose Verification means that the repeated experiments with minimum verification index of 90% and maximum of 97% provide acceptable results.

Block	Parameter	Value
Edge Detection	t_h: high threshold	$max(E_s)$
	t_l: low threshold	$0.4t_h$
Edge Smoothing	d_t: smoothing threshold	(9, 11)
Segmentation	n: number of points in each segment	(45, 55)
Segment Verification	h_d: distance histogram threshold, equation (22)	(2, 5)
	ω_c: discontinuity threshold, equation (23)	(5, 8)
	number of bins of a histogram	(5, 9)
Pose Verification	verification threshold (%)	(90, 97)

4.5 Discussion: Comparison Results

The proposed method has been compared with adaptive random walk (ARW) map merging [4]. ARW is based on search and verification. In ARW, the similarity indices of two maps are calculated given a set of transformations. The set is composed of known rotations and translations. The transformation corresponding to the best similarity index is selected as the start point of an adaptive random walk search to merge the maps more accurately. Finding the initial transformation from the set takes the majority of the processing time. To compare the results with ARW, a faster version of ARW is used where a set of 144 transformations have been investigated. In this case the proposed algorithm runs at least ten times faster.

Table 2 summarizes the comparison of the processing times for the two experiments. To perform an accurate comparison, all computations were performed on a Core2Duo 2.66 GHz laptop. In the first experiment, the algorithm takes about 84 seconds to run. This is about half of the time that the algorithm presented in [4] required to run on the identical environment producing similar results. For the second experiment, the entire algorithm required about 95 seconds to run, which is less than the time required for ARW.

Table 2: Comparison of Processing Times

Method	Experiment 1	Experiment 2
Map segmentation	$\approx$ 84 sec	$\approx$ 95 sec
ARW map merging [4]	$\approx$ 152 sec	$\approx$ 160 sec

Table 3 compares verification indices of the these two methods for the two experiments. In the first experiment the indices are close but in the second experiment, the proposed method provides a better verification index.

Table 3: Comparison of Verification Indices

Method	Experiment 1	Experiment 2
Map segmentation	$\approx$ 95 %	$\approx$ 98 %
ARW map merging [4]	$\approx$ 95 %	$\approx$ 94 %

In experiment 1 the amount of overlap was approximately 35%, in experiment 2 the average overlap of the original maps was approximately 25% . However, it is important to note that success or failure of the map merging process is more dependent on the existence of matchable segments within the overlapping sections of the map than on the actual size of overlapping sections of the maps.

According to the size and processing time of the maps in experiment 1 and experiment 2, the time complexity is $\mathcal{O}(n_1 n_2)$ where n_1 and n_2 are the size of each of the two maps being merged. Space complexity of the algorithm, which is the required memory, grows quadratically, as the number of the cells in the maps grow quadratically. The size of the selected segments also affect the processing time. If the size of the segments are smaller than the typical values, introduced in Table 1, then more processing time is required. If the size of the segments are larger than the introduced values, then the algorithm might fail to find the matching segments. Variation of the time complexity of the proposed algorithm for the given values of the segment size in Table 1 is negligible.

The feature extraction method proposed here based on segment matching is particularly well-suited to matching occupancy grid map structures. Other feature extraction methods designed to work on imagery, such as Fourier transform, were tested and produced poorer results.

It should be stated that, in this work, the paths of the robots were planned *a priori* with the real focus of the approach being on map merging. In future studies we will explore how to combine the proposed map merging approach with path planning for efficient real time exploration and localization.

In addition, in this work maps are aligned only once to find the relative transformation between the two robot poses. Once this transformation is known, it is assumed that further explored parts of the map can be easily fused. It would be interesting to explore how the performance could be improved if the map merging process happens repeatedly as the robots explore the environment.

5 Conclusion

In this paper, a new method of multiple robot SLAM is developed. The proposed method is based on improvements made to single robot SLAM algorithms. Results are verified with tests performed in real environments. Novel aspects of this approach include, preprocessing of occupancy grid maps using Canny Edge detection and smoothing, segmentation using a novel histogram method to find unique geometrical characteristics, cross correlation to find matching patterns in maps, methods for determining approximate and exact transformation matrices that relate the maps, and verification. A major advantage of this approach is that there is no need for the robots to meet one another to determine their relative positions. They can do localization and mapping independently without relying on tracking one another.

In the future, improving the pose extraction by using better search algorithms will be investigated. Incorporating tracking information can also improve the results, meaning that if the robots do happen to see each other, they can find their relative poses quickly and easily and use them to generate more accurate results.

Acknowledgement

This research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Canada Foundation for Innovation. The authors would also like to thank the anonymous reviewers for improving this manuscript.

References

[1] S. Saeedi, L. Paull, M. Trentini, & H. Li, Multiple Robot Simultaneous Localization and Mapping, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, 853-858.

[2] R. Aragues, J. Cortes & C. Sagues, Distributed consensus algorithms for merging feature-based maps with limited communication, *Robotics and Autonomous Systems*, 59(3-4), 2011, 163-180.

[3] T. Bailey, *Mobile robot localization and mapping in extensive outdoor environments*, doctoral diss., University of Sydney, Sydney, NSW, Australia, 2000.

[4] A. Birk & S. Carpin, Merging occupancy grid maps from multiple robots, *Proceedings of the IEEE: Special Issue on Multi-Robot Systems*, 94(7), 2006, 1384-1387.

[5] S. Carpin, A. Birk & V. Jucikas, On map merging, *Robotics and Autonomous Systems*, 53(1), 2005, 1-14.

[6] A. Censi, An ICP variant using a point-to-line metric, *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2008, 19-25 .

[7] E.R. Davies, *Machine vision, theory, algorithms, practicalities, 3rd Edition*, (Morgan Kaufmann, 2005).

[8] A.J. Davison & D. W. Murray, Simultaneous localizationand map-building using active vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7), 2002, 865-880.

[9] A. Elfes, Occupancy grids: A stochastic spatial representation for active robot perception, *Sixth Conference on Uncertainty in AI*, 1990, 7-24.

[10] A. Eliazar & R. Parr, DP-SLAM 2.0, *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2004, 1314-1320.

[11] C.M. Gifford, R. Webb, J. Bley, D. Leung, M. Calnon, J. Makarewicz, B. Banz & A. Agah, A novel low-cost, limited-resource approach to autonomous multi-robot exploration and mapping, *Robotics and Autonomous Systems*, 58(2), 2010, 186-202.

[12] A. Gil, O. Reinoso, M. Ballesta & M. Julia, Multi-robot visual SLAM using a Rao-Blackwellized particle filter, *Robotics and Autonomous Systems*, 58(1), 2010, 68-80.

[13] G. Grisetti, C. Stachniss, & W. Burgard, Improved techniques for grid mapping with Rao-Blackwellized particle filters, *IEEE Transactions on Robotics*, 23(1), 2007, 34-46.

[14] J. Guivant & E. Nebot, Optimization of the simultaneous localization and map-building algorithm and real-time implementation, *IEEE Transactions on Robotics and Automation*, 17(3), 2001, 242-257.

[15] D. Hahnel, W. Burgard, D. Fox, & S. Thrun, An effcient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, 206-211.

[16] A. Harati & R. Siegwart, A new approach to segmentation of 2D range scans into linear regions, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, 2083-2088.

[17] J. Hawkins, *On Intelligence*, (Times Books, 2004).

[18] A. Howard, Multi-robot simultaneous localization and mapping using particle filters, *International Journal of Robotics Research*, 25(12), 2006, 1243-1256.

[19] L. Jaulin, A nonlinear set membership approach for the localization and map building of underwater robots, *IEEE Transactions on Robotics*, 25(1), 2009, 88-98.

[20] A. Koenig, J. Kessler & H.M. Gross, A graph matching technique for an appearance-based, visual SLAM-approach using Rao-Blackwellized particle filters, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, 1576-1581.

[21] K. Konolige, D. Fox, B. Limketkai, J. Ko & B. Stewart, Map merging for distributed robot navigation, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, 212-217.

[22] K. LeBlanc & A. Saffiotti, Multirobot object localization: a fuzzy fusion approach, *IEEE Transactions on Systems, Man and Cybernetics-Part B*, 39(5), 2009, 1259-1276.

[23] F. Lu & E.E Milios, Robot pose estimation in unknown environments by matching 2D range scans, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1994, 935-938.

[24] P.M. Newman, J.J. Leonard & R.J. Rikoski, Towards constant-time SLAM on an autonomous underwater vehicle using synthetic aperture sonar, *Proceedings of the Eleventh International Symposium on Robotics Research*, 2003, 409-420.

[25] V. Nguyen, D. Fox, A. Martinellii, N. Tomatis, & R. Siegwart, A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, 1929-1934.

[26] J. Nieto, T. Bailey,& E. Nebot, Recursive scan-matching SLAM, *Robotics and Autonomous Systems*, 55(1), 2007, 39-49.

[27] R. Johann, On the determination of functions from their integral values along certain manifolds, *IEEE Transactions on Medical Imaging*, 5(4), 1986, 170-176.

[28] T.J. Ross, Fuzzy logic with engineering applications, (Wiley, 2004).

[29] M. Smith, I. Baldwin, W. Churchill, R. Paul & P. Newman, The new college vision and laser data set, *The International Journal of Robotics Research*, 28(5), 2009, 595-599.

[30] R. Smith, M. Self & P. Cheeseman, A stochastic map for uncertain spatial relationships, *Fourth International Symposium of Robotics Research*, 1987, 467-474.

[31] E. Stump, V. Kumar, B. Grocholsky & P.M. Shiroma, Control for localization of targets using range-only sensors, *The International Journal of Robotics Research*, 28(6), 2009, 743-757.

[32] S. Thrun, Learning occupancy grids with forward sensor models, *Autonomous Robots*, 15, 2002, 111-127.

[33] S. Thrun, W. Burgard & D. Fox, *Probabilistic robotics*, (Cambridge, MA, The MIT press, 2005).

[34] S. Thrun & Y. Liu, Multi-robot SLAM with sparse extended information filers, *Robotics Research*, 15, 2005, 254-266.

[35] I. Ulrich & I. Nourbakhsh, Appearance-based place recognition for topological localization, *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2000, 1023-1029.

[36] M. Wang & C.H. Lai, *A Concise Introduction to Image Processing*, (CRC Press, 2008).

[37] H.D. Whyte & T. Bailey, Simultaneous localization and mapping (SLAM): part I the essential algorithms, *IEEE Robotics and Automation Magazine*, 13(3), 2006, 108-117.

[38] X.S. Zhou & S.I. Roumeliotis, Multi-robot SLAM with unknown initial correspondence: the robot rendezvous case, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, 1785-1792.

[39] S. Thrun, A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots, *The International Journal of Robotics Research*, 20(5), 2001, 335-363.